DOCUMENT RESUME

ED 290 452 IR 013 108

AUTHOR Nachmias, Rafi; And Others
TITLE Acquisition of Basic Computer Programming Concepts by Children. Technical Report No. 14.
INSTITUTION Tel-Aviv Univ. (Israel). Computers in Education Research Lab.
SPONS AGENCY Israel Ministry of Education, Jerusalem.
PUB DATE Feb 85
NOTE 37p.; For related reports, see IR 013 109-115.
PUB TYPE Reports - Research/Technical (143)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS Age Differences; Analysis of Variance; Attitude Measures; *Cognitive Processes; Computer Literacy; Concept Formation; Foreign Countries; *Instructional Development; Intermediate Grades; *Microcomputers; Preadolescents; *Programing; Programing Languages; Psychological Studies; Sex Differences; *Student Attitudes; Units of Study
IDENTIFIERS Basic Programing Language; LOGO Programing Language

ABSTRACT

The process through which the basic principles and concepts of computer programming language are acquired by children was investigated via the development and testing of a teaching unit for fourth and sixth grade students. This unit had three components: (1) basic programming without variables; (2) basic programming with variables; and (3) integration of concepts and simple programs into a combined project and presentation of the principles of top-down design. Two studies were undertaken, one in a laboratory setting and the second in a classroom setting. The high variability of achievement by the students was tested and found to be explained by grade level (age) and academic achievement, not gender. It was also found that the data supported the contention that basic computer programming can be introduced at the upper elementary level, but that the introduction of more abstract concepts may need to be postponed to a later stage of learning. Additionally, it is noted that the teaching of structured programming may not necessarily be relevant to the teaching of introductory programming to children. A 25-item bibliography is provided. (EW)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

THE COMPUTERS IN EDUCATION RESEARCH LAB.

המעבדה לחקר יישומי מחשבים בחינוך

ED290452

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy

ACQUISITION OF BASIC COMPUTER PROGRAMMING CONCEPTS BY CHILDREN

Rafi Nachmias, David Mioduser, David Chen

Technical Report No. 14

February 1985

IR013108

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

David Chen

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

TEL AVIV UNIVERSITY SCHOOL OF EDUCATION

בית הספר לחינוך

אוניברסיטת תל אביב

UNIT FOR COMMUNICATION & COMPUTER RESEARCH IN EDUCATION

היחידה לחקר תקשוב בחינוך

THE COMPUTERS IN EDUCATION RESEARCH LAB

המעבדה לחקר יישומי מחשב בחינוך

ACQUISITION OF BASIC COMPUTER PROGRAMMING CONCEPTS BY CHILDREN

Rafi Nachmias, David Mioduser, David Chen

Technical Report No. 14

February 1985

(Submitted to the "Journal of Educational Computing Research")

This paper has been prepared at the Computers in Education Research Lab.; The Unit for Curriculum R&D, School of Education, Tel Aviv University, with assistance of the Israel Ministry of Education. The authors wish to thank the team members that helped at different stages of this work.

School of Education Tel Aviv University Tel Aviv 69978 Israel

BEST COPY AVAILABLE

1. INTRODUCTION

The major goal of the present study is further clarification of the process through which basic principles and concepts of programming language are acquired by children. Microcomputers are becoming increasingly popular in both homes and schools, and a growing number of children are thus learning introductory programming courses, either in the formal or the informal education system. Therefore, educators are much concerned with questions such as: Should programming language be taught at school? What are the goals of teaching programming? How should teaching proceed in order to best achieve these goals and how much time is needed for that? Which language should be taught? Which contents should be taught, and in what sequence? and, finally, which is the best age to start teaching programming? If these questions are to be answered, more information regarding the process of learning a programming language by children and novices, is necessary. Such information may contribute to the improvement of the quality of computer programming curricula, which are currently being developed at an accelerated rate (Moshell, 1982; Luehrmann & Peckman, 1984), as well as deciding on the principles of new programming languages and integrated programming environemnts (diSessa, 1984).

A considerable number of the children learning a programming language are faced with difficulties in the process of learning. These are caused by basic misunderstanding of concepts of programming, and the lack of suitable curriculum for children. Empirical data regarding the learning process of a programming language by the naive learner may contribute greatly to the formulation of a realistic picture of these difficulties and misconceptions, thus enabling to improve the teaching of programming to beginners (Shiel, 1981). Computer programming is a complex cognitive activity, comprised of a very large number of basic skills, which must all be taken into consideration when deciding what to teach the naive programming student (Shiel, 1981). The present study has attempted to cope with this complex problem, by dividing the programming activity into specific skills, and examining how they are learned by children in grades 4 and 6. The detailed list of the concepts on which we focused is given in chapter 2.

A learning unit for teaching the basic concepts of a programming language to children, employing the BASIC language, was developed at the COMPUTER IN EDUCATION RESEARCH LAB. for the purpose of the present study. This Unit is one component of a larger curriculum in Computer Literacy, intended for the upper grades of elementary school and Junior High school, as part of the technological-scientific basic education (Nachmias, Mioduser and Chen, 1985). Most students will not end up as professional programmers, but they will be computer-users, living in a computer-based society. The summary of the principles on which the learning unit is based appears in Chapter 3.

The lack of adequate empirical body of knowledge dictated that the study be carried out in two stages. The first comprised of 6 case-studies, and included the collecting of detailed qualitative data about single children, learning a programming language; In the second stage the learning unit was implemented in the school environment, and the trends revealed in the first stage were examined in a larger sample.

The goals of the study were:

1. Examination of children's ability to understand and apply (at varying levels) basic concepts of a programming language.
2. Examination of the differences between children in grades 4 and 6, with respect to their ability to understand and apply the various concepts, as well as the time required for learning these concepts.
3. Indicating and mapping the misunderstandings, misconceptions and difficulties encountered by children while learning programming.
4. Examination of the influence of academic achievements and the student's sex, on achievements in programming.
5. Examination of curricular aspects on the manner in which individuals, and entire classes, learn programming.

2. BASIC CONCEPTS OF PROGRAMMING LANGUAGE

Computer programming is a complex activity, requiring a wide range of motor and cognitive skills (Weinberg, 1977; Shiel, 1980). The difficulties encountred by novices, arise mostly from the need to activate simultaneously a large number of basic skills, some of which already exist in the repertoire of the learner, and some of which still need to be learned. Shneiderman (1980) differentiates between syntactic knowledge and semantic knowledge, and suggests that such knowledge be organized in a spiral form (Shneiderman, 1977). Mayer (1979) proposed a subdivision of the BASIC language into 8 levels, from the single statement level and up to the computer program level.

Similarly, We suggest that the skills and basic concepts required of the naive programmer be divided into five knowledge domains:

1. The techincal skills of using the keyboard and mastering the technology.
2. Knowledge of the syntax of the various statements.
3. Understanding the execution flow of the program.
4. Understanding the changes of values of the variables during the execution of a program.
5. Task-specific knowledge.

We believe that learning a new concept or application of a programming language involves two types of learning: The first is related to increasing the knowledge and understanding within each of the above knowledge domains; the second requires integration of these various domains. For example: In order to write a program which prints consecutive numbers in a loop, the programmer should be able to type; he should know the syntax of the statements, understand the principle of repetition, understand the concept of variable, and master the arithmetic operation of addition. Each of the above skills represents one of the above-mentioned knowledge domains. However, to be able to write such a program, and understand how it is being executed, the writer must also integrate these areas.

For example, he should understand the changing of the variable's values in each iteration of the loop, until its completion. Therefore, if programming teaching is to be effective, the two types of learning should be combined, through problem solving and application activities. Throughout the present study, we have examined the learning of the concepts at each knowledge domain separately, as well as the attempts of the students to integrate the various domains.

The concepts employed in the present study were:

1. Parametric statements and basic computer graphics.

1.1 Perception of the computer screen as Cartesian coordinates

1.2 Ability to locate points and lines on the screen.

1.3 Syntax of basic graphic statements.

1.4 Understanding of the concept of parameter (statements with one parameter and multiple-parameter statements).

2. The computer program.

2.1 Understanding the concept of a computer program as a series of commands written in a specifically designed computer language, which are recorded in the computer's memory.

2.2 Understanding the order in which commands are being executed.

2.3 Controlling the program through the use of the RUN, LIST and NEW commands.

2.4 Ability to debug simple computer programs.

3. Random number generation.

3.1 Understanding the concept of random number and the range of numbers from which it is generated.

3.2 Changing the size of the group of numbers from which a random number is generated.

3.3 Controlling the starting point of the group from which a random number is generated.

3.4 Employing random numbers as parameters in graphic and other statements.

4. Infinite loop

4.1 Understanding the concept of loop as a series of statements which are executed repeatedly.

4.2 Understanding the implications of the absence of stopping rule.

4.3 Understanding the use of GOTO statement for building loops

4.4 The ability to stop an infinite loop using the keyboard.

5. Variable

5.1 Understanding the concept of variable as a location in the computer's memory, which is given a name and used to store alphanumeric information.

5.2 Using the variable as a parameter in programming statements.

5.3 Using the variable as a numerator.

5.4 Using the variable as a cumulative numerator in the formation of series of numbers.

6. Input

6.1 Understanding the role of input in the performance of a program.

6.2 Inputting through stopping the program using INPUT statement.

6.3 Inputting during the program execution using peripheral equipment.

7. Conditioning

7.1 Understanding the concept of conditioning.

7.2 Understanding the logical operations which may be used with the IF-THEN statement.

7.3 Using the IF-THEN statement to stop the program.

7.4 Using the IF-THEN statement for branching a computer program.

Each of the above concepts comprises a chapter in the learning unit.

3. METHOD

The Learning Unit:

The learning unit for teaching programming language to children, which served as the major tool for instruction in the present study, was composed of three parts: The first is basic programming, without variables; the children learn to write single-loop graphic and textual programs, performing simple tasks. The second part includes basic programming with variables, and it is centered around the concept of variable, and the input and conditioning statements. The third part is comprised of integrating the concepts and simple programs into a combined project, and presenting the principles of top-down design.

The emphasis, while developing the unit, was on the following principles:

1. Hierarchical presentation of the material in all five knowledge domains; each new concept or skill was taught on the basis of components of knowledge which were learned earlier, and mastered.
2. An attempt was made to teach concurrently the knowledge of each of the five domains, while trying to integrate the various levels through application and problem-solving activities.
3. The areas in which the student is requested to apply the newly acquired knowledge are variegated, including graphics, alphanumeric text and sound.
4. The learning is accompanied by active work with micro-computers from the very beginning, and for most of the learning period.
5. An attempt was made to disconnect as much as possible the programming teaching from any mathematical contents which are not directly relevant to programming.
6. The learning material is composed of a variety of programming language tasks, intended for students at different levels and with differing abilities, so as to ensure effective and interesting learning for the majority of the students.

Further information concerning the learning unit, its principles, the organization of its contents

and the teaching design, appears elsewhere (Nachmias, Mioduser and Chen, 1985a).

Subjects:

In both studies, the subjects were 4th and 6th grade students, with no previous knowledge of programming. For the first study, 6 children (3 in grade 4 and 3 in grade 6) with higher than average intelligence (but not reaching the "gifted" category) were selected. A school in an established neighbourhood was selected for participation in Study 2. Two grade 4 and two grade 6 classes comprised the subject population in this study (n=111). A selection questionnaire served to exclude any students in these four classes with previous knowledge in programming, or those who have participated in the past in an Introductory Programming course. The total number of excluded students was 29 (6 4-graders and 23 6-graders). Another 9 students did not take part in the study (1 4-grade and 8 6-grade pupils). The final number of subjects participating in Study 2 was thus 73 (45 in the 4th grade and 28 in the 6 grade).

Procedure:

The need to reconcile the conflict between reliability and generalizability, as described by Shiel (1981), dictated the use of two different designs:

Study 1: An exploratory study, comprised of 6 case-studies (three 4-grade pupils and three 6-grade students), who learned a programming language under laboratory conditions. Lessons were given twice weekly, approximately 90 minutes each time, for about four months. A total of 25 double-lessons were given. Two students participated in each meeting, at whose disposal were micro-computers and learning materials. Each student had an instructor, whose responsibility was to explain unclear points, add tasks, raise problems which were encountered throughout the learning, and help in deciding on the course of learning. Another observer documented in detail the interaction of the student with the learning material, and the difficulties and misconceptions encountered during the

learning. The nature of the process was of self-learning, guided by the learning unit. The last lessons were devoted to summarizing projects, which included the entire scope of learning; finally, a concluding test was administered.

Study 2 was conducted in four school classes (two 4th grade and two 6th grade), immediately following the scheduled school work. Twelve weekly lessons were given, each of 2-hour duration. Each lesson began with a frontally presented explanation, accompanied with a demonstration on the classroom computer. The students then turned to practice with the microcomputer (Apple IIE), two students working with one computer. Following the teaching of each concept, the students' knowledge was examined; this was done through a battery of specifically-designed tests. During the last lesson, a final personal interview in front of the computer was administered to each student. An attitude questionnaire was also completed.

The sequence of teaching and testing was as follows:

Selection test

Lesson 1: Graphic statements

Lesson 2: The computer program

Test no. 1

Lesson 3: Writing a concluding graphic project

Test no. 2

Lesson 4: Random number generation

Lesson 5: Infinite loop

Test no. 3

Lesson 6: Applications of loop (simple animation)

Lesson 7: Printing of text and calculations

Test no. 4

Lesson 8: Variable

Lesson 9: Applications of variable.

Lesson 10: Input statement

Test no. 5

Lesson 11: IF statement

Lesson 12: Summarizing Project

Final test and attitude questionnaire.

Data collecting instruments:

In Study 1, a clinical protocol, recording the learning process, served to collect most data. Towards the conclusion of the experiment, a final test was conducted, which included the entire scope of learning. There was no time limit for completing the test, which, in effect, lasted 60-80 minutes.

The test examined the understanding and application ability of 5 groups of concepts which had been learned: (1) Low-resolution graphics and parametric statements; (2) Random number and infinite loop; (3) The use of random numbers and loops in printing statements; (4) the concept of variable; (5) Input statement and conditioning.

For each concept, the understanding and application ability were tested with five questions, comprising of:

1. Identifying syntax errors;
2. Identifying the function of single statements;
3. Identifying the result (or product) of running a program;
4. A slight change of a given program;
5. Writing a short program employing the concepts.

The description of programs included in the test is given in Table 3.1

TABLE 3.1: DESCRIPTION OF THE FINAL TEST OF STUDY 1

application level / concepts	1. Locate and correct syntax errors	2. Which part of the program is executed by each statement?	3. What appears on the screen following the running of the program?	4. Change the program so that:	5. Write your own program:
basic computer graphics	10 GR 20 COLOR 4 30 PLOT 7 40 PLOT 5,5 TO 10,10 50 HPLOT 5,5 TO 10,10 60 HLIN 5,5 AT 10	10 GR 20 COLOR= 15 30 VLIN 10,20 AT 15 40 PLOT 15,25		draw the same form in high-resolution graphics	Drawing a rectangle in high-resolution graphics
random number and infinite loop	5 GR 15 COLOR= RND * 16 20 PLOT RND (1) * 40 30 GOTO 10	10 GR 20 COLOR= RND (1) * 16 30 PLOT RND (1) * 10, RND (1) * 10 40 GOTO 20		the result appears on half the screen	Drawing a simple form which changes colors
print	10 PRINT "SHALOM" 20 PRINT 2" + "3 30 PRINT 12; 40 PRINT "9+3" = :	10 REM GAME FOR TWO 20 PRINT "PLAYER 1" ,"PLAYER 2" 30 PRINT RND (1) * 10, RND (1) * 10 40 PRINT "————————————————" 50 PRINT 60 FOR T = 1 TO 2000: NEXT T 70 GOTO 30		the game includes three players; the delay between games will be longer and each time the output appears at the top of the screen.	Imitating the throwing of two coins, presenting the results on the screen
variable	10 X = 10:X = 8 20 AB = 30 30 X = X + A 40 X$ = A	10 GR 20 COLOR= 5 30 PLOT X,X 40 X = X + 1 50 GOTO 30		a horizontal line is drawn, composed of dots separated by two units.	Drawing a randomly located line, of 50-dot length
input + if	10 INPUT 6 20 INPUT X + 1 30 IF INPUT X = 4 THEN PRINT "X=4" 40 IF X > 30 GOTO 10 50 IF X + 6 THEN GOTO 20	10 INPUT X 20 IF X = 999 THEN GOTO 100 30 A = A + X 40 GOTO 10 100 PRINT 110 PRINT A		so that inputting 10000 results in printing -END-	Which requires inputting two numbers: if equal, print their sume; if unequal, print their product

The data in Study 2 were collected through written tests (comprised of multiple-choice and open questions), and personal interviews in front of the computer. Exemplary test forms are included elsewhere (Nachmias, Mioduser and Chen, 1985b).

Test no. 1: Examined the comprehension of basic low-resolution graphic statements: Three questions presented either a dot or a line, and the student was asked to write the statement which was needed to draw it; three other questions included a statement, and the student was requested to draw the dots or lines on a given schematic computer screen.

Test no. 2: Was comprised of an interview in front of the computer, intended to examine the student's mastery of a computer program. The examiner asked the student to run a program, present the statements on the screen, add and delete statements, change a statement and erase the program from the computer's memory.

Test no. 3: Comprised of a multiple-choice questionnaire dealing with random numbers. The items were related to the group of numbers from which a random number is generated, the syntax of the function, generating a random number, employing this function as a parameter in a graphic statement (drawing a dot), predicting the product of a program which includes random numbers.

Test no. 4: Multiple-choice questionnaire dealing with infinte loop. Items were related to the comprehension of the concept of loop in a series of repeated statements; comprehension of stop rule; the syntax of the statement and stopping a loop using the keyboard.

Test no. 5: Was composed of multiple-choice and open questions, dealing with the variable. Most questions involve the identification of the output (text and graphic) of a given program, and writing a simple program introducing values into two variables and printing their sums.

Final Test: This was a personal interview in front of the computer, which lasted approxiamtely 15 minutes. The examiner rated the student's performance on a given form. It should be emphasized that all these tasks have been referred to during the learning, and none was entirely new to the student. Each student was asked to complete the following tasks:

- Write a program which draws a dot on the screen, run this program, list its statements.

- Change the color of the dot to a random color, change a statement so that the location of the dot will be determined randomly.
- Add a statement in order to fill the screen with random dots in various colors (infinite loop); change the location to which the loop returns.
- Predict what will be performed by a short program with variables, which is loaded from a diskette; change a statement in order to receive a different output.
- Write a program which receives a number as input and multiplies it by 7.
-Add a statement to test whether the number is equal to 0; if it is, print the word "zero".

4. RESULTS

STUDY 1:A CLOSE LOOK AT CHILDREN'S ACQUISITION OF PROGRAMMING: SIX CASE STUDIES

The data collected in Study 1 are qualitative, and are derived from detailed documentation of the learning process of the participating children. The following is only a representative impression of the children's work and their success in applying the learned concepts, as well as the difficulties encountered by them during learning. A detailed description of the results of Study 1 appears elsewhere (Nachmias, Mioduser & Chen, 1984).

The mastery of the technical skills of using the computer (such as using the keyboard, and, at a later stage, employing peripheral devices) was achieved very quickly by the 6 children participating in this experiment. This was not surprising, in view of our previous experience. It is also in agreement with reports found in the literature, concerning the facility of use of the computer by children. The fourth grade students participating in this study did not have any previous knowledge of English, and were not acquainted with Latin alphabeth (English studies in Israel do not start until the fifth grade); we therefore expected some difficulties in identification of letters and statements. However, we were surprised at the speed with which these children acquired the use of a

symbol system that was foreign to them, and the comprehension of statements, the literal meaning of which was unknown.

The following is a brief description of the process of acquisition of the various concepts:

Graphics and parametric statements

The learning of the syntax of execution statements with parameters (graphic statements at the beginning of the course and other statements later on) was very fast, and following 3-5 hours of teaching, the children were able to run simple, though not short, linear programs. These projects were varied with regard to both color and form, indicating a high ability of planning and capability of carrying out such plans with the use of a computer program. The only differences which were noted between children at the two age levels were the speed of performance (6-graders were quicker) and the frequency of typing errors, which was higher among 4-grade pupils.

Another interesting phenomenon was the increasing difficulty of the task, as a function of the number of parameters in the statement. There were practically no errors in single-parameter statements (such as color statement); a few errors were made in two-parameter statements (dot drawing statement), and the highest number of errors was evident in multiple-parameter statements (line drawing statement). This trend was noted along the entire process of learning.

During the first lessons the children experienced the transition from a state of total ignorance, to a state of mastery of the computer; the output received was indeed impressive (low-resolution graphics color drawings). This transition in the control of the computer was exciting, and induced both enthusiasm and high motivation for the continuation of learning.

The 6-grade children demonstrated curiosity for learning additional statements and applications, whereas the 4-grade pupils tended to continue drawing, using those statements they were already familiar with.

Random number generation:

Many difficulties were caused by the use of the function for creating a random number, as a parameter in multiple-parameter graphic statements (e.g. dot or line). On the other hand, employing a single-parameter statement (selecting a random color) did not cause any problems. A typical error was drawing a dot at a random location, writing the statement PLOT RND(1)*10 (instead of: PLOT RND(1)*10,RND(1)*10). All children understood the changing of the range of numbers from which the random number was to be selected; however, adding a fixed factor to the random number (e.g. RND(1)*10+5).

Infinite loop:

The idea of performing the same part of the program several times (through the use of GOTO statement) was basically well understood. Yet, a number of major misconceptions were repeatedly noted in both age groups. A frequent error was the lack of awareness of the fact that statements cannot be typed while the infinite loop is running. Children therefore tried to type statements while the program was being executed, since they were convinced that the statements are accepted by the computer. Another error which occurred rather often was adding statements to the program after the GOTO statement (statements which will never be executed by the computer), on the assumption that the very inclusion of a statement in the program ensures its execution. On other occasions, the relationship between the GOTO statement and the infinite repetition of a part of the program, was not understood. Some children thought that writing a statement for drawing a randomly placed dot is responsible for the repetition of the statement, regardless of the GOTO statement.

Variable:

There are two stages in understanding the concept of variable, which differ from each other drastically: The first stage presents the variable as a location in the computer's memory, which has alphanumeric contents and may be assigned a name. The second stage involves manipulating the value within the variable (e.g. X=X+1). All children appeared to understand the first stage; the second

stage, however, was practically incomprehensible to the 4th grade children, whereas it was well understood by the 6th grade pupils, who managed to apply this knowledge later, in the final programs written by them (for example: adding numerator to a program, or employing the principle of changing the variable's value for simple animation).

Input:

This concept was presented as entering a value into the variable while the program is running. Here, too, the 4th-grade children appeared to demonstrate limited understanding, and even more limited ability to apply this concept. A typical difficulty was the inability to recognize whether the program is at the stage of waiting for input, or whether it has been completed. One such example is writing a new statement in response to the question mark of the input statement, instead of supplying the numerical input. At a later stage of the learning, inputting via peripheral input devices, such as joy-stick or pedal was introduced. This attempt was successful, and seemed to clarify the input concept and the changing of the variable's values.

Conditioning:

The difficulties encountered by the children (especially the 4th grade students) derived from misunderstanding the logical aspects of the conditioning. In both age groups, severe problems were encountered when employing the IF statement for branching a program.

The final test:

As has been mentioned previously, a comprehensive final test was administered at the end of learning. The results of this test are presented in Table 4.1. It shows the mean scores of the three 4th grade and three 6-grade students, in five concepts which were tested. For each such concept, five questions, representing five levels of difficulty, were given. The maximal score in each cell is 20, and the maximal total score for each concept is 100. (The detailed description of the test is given in Chapter 2).

TABLE 4.1: MEAN SCORES IN THE FINAL TEST OF STUDY 1

concept	question	4th grade	6th grade
basic computer graphics	1	12	20
	2	20	20
	3	17	19
	4	13	17
	5	7	15
	all	69	91
random number and infinite loop	1	14	16
	2	19	20
	3	15	17
	4	10	17
	5	13	15
	all	71	85
print	1	3	17
	2	13	18
	3	12	17
	4	5	17
	5	1	17
	all	34	86
variable	1	10	15
	2	10	19
	3	0	17
	4	0	12
	5	1	6
	all	21	69
input + if	1	0	16
	2	0	20
	3	0	14
	4	0	10
	5	0	5
	all	0	65

Several trends are evident in the above findings:

A. The achievements of the 4th grade students with regard to the first two concepts are good. A significant decrease is seen in the comprehension of printing of random numbers, which is the first to include some use of variables. Programming tasks which include working with variables, reveal that 4th grade students did not understand these concepts at all.

B. The 6th grade students revealed high achievements in understanding the first three concepts, and the decrease in level of understanding of the variable concept is mild.

C. There are obvious differences between children in the 4th and the 6th grade, with regard to the level of comprehension and ability to apply all concepts. This difference increased with learning,

and was more evident when variables were used.

D. Independent writing of a program, or introducing changes into the program, incurred most difficulties, especially with the younger children. On the other hand, understanding a single statement and identifying its effect on the prgoram, as well as identifying the output of an entire program, were easier and achieved with greater success.

STUDY 2: TEACHING PROGRAMMING LANGUAGE IN CLASS

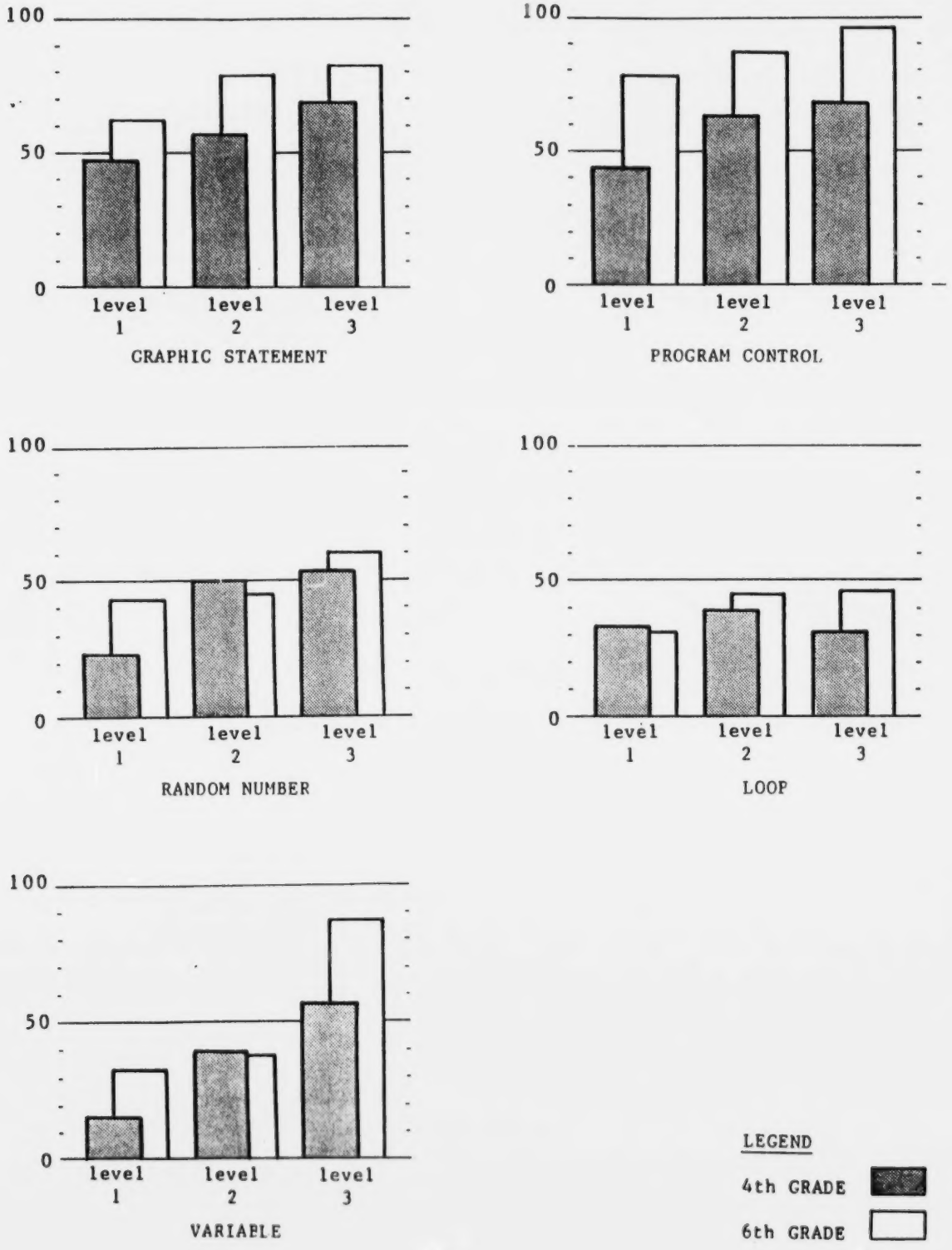
This study was intended to further examine some of the trends identified in Study 1, under school conditions. Table 4.2 summarizes the achievements of the students in the five tests conducted during the learning. The scores are the mean number of correct answers in each test, expressed as percents. Data are presented for the 4th and 6th grades, and divided, at each grade level, according to academic achievements. Figure 1 presents the same data graphically.

TABLE 4.2: THE MEAN SCORES IN THE 5 INTERMEDIATE TESTS ADMINISTERED IN STUDY 2

	test 1: Graphic statements (6 items)				test 2: The program (6 items)				test 3: Random number (5 items)				test 4: Loop (4 items)				test 5: Variable (12 items)			
grade level	1	2	3	ALL	1	2	3	ALL	1	2	3	ALL	1	2	3	ALL	1	2	3	ALL
4	47 (12)	57 (14)	69 (19)	60 (45)	44 (10)	63 (7)	68 (17)	60 (36)	23 (12)	50 (14)	53 (19)	44 (45)	33 (12)	39 (14)	31 (18)	35 (44)	16 (8)	39 (13)	57 (19)	43 (40)
6	62 (8)	79 (11)	83 (7)	57 (26)	78 (6)	87 (9)	96 (7)	88 (22)	43 (7)	45 (11)	60 (7)	49 (25)	31 (8)	45 (11)	46 (7)	41 (26)	33 (6)	38 (7)	87 (6)	52 (19)
all				65 (71)				71 (58)				45 (70)				37 (70)				46 (59)
significance		*		*	**	*	**	**	*										*	

* P<.05
** P<.01

FIG. 1: MEAN SCORES IN THE FIVE INTERMEDIATE TESTS (STUDY 2)



The data presented in Table 4.2 and the results obtained for single items on each test, reveal the following :

1. Graphics and parametric statements:

The 6th grade students, especially the better ones, demonstrated high achievements. There are significant differences between students of 4th and 6th grades. Examination of single items reveals two trends: The statement for drawing a dot (two-parameter statement) was far better understood than the statement for drawing a line (a three-parameter statement): 75% answered the items concerning dots correctly, as compared to only 46% correct answers for the line drawing items. The other trend showed that drawing on the basis of a given statement was much more difficult than writing the statement to draw the same dot (55% of the students were correct in the former, as compared to 75% correct answers for the latter).

2. Program

There are significant differences between students of the 4th and 6th grades. Students in the 6th grade have very good achievements, indicating undestanding and mastery of the concept of program. The achievements of the 4th grade students, on the other hand, are average.

3. Random number

Thirty three percent of the students made mistakes in questions related to identifying the range from which a random number is generated. When comprehension of random numbers had to be integrated with writing a statement for drawing a point (PLOT RND(1)*40, RND(1)*40), the percentage of errors was as high as 75%.

4. Loop

Even the good students in the 6th grade did poorly with respect to the concept of loop. Examination of single items demonstrates that items testing the role of GOTO in the program resulted in very

poor achievements (the average number of those replying correctly was 18%). One item presented a program in which a statement appeared after the infinite loop. The students were asked whether this statement would be executed; 79% gave the wrong answer!! There was no significant difference between 4th and 6th grade students with regard to the above. The only item to which correct answers were given with high frequency (76%) was related to stopping an infinite loop through the keyboard.

5. Variable

Unexpectedly, the variability among students regarding the understanding of the variable concept may be explained by academic achievements, more than it is explained by age. Students with average or weak academic achievements, in both grade levels, did poorly, as opposed to particularly high achievements of the good students in grade 6. The latter reached 87% correct answers. The significant difference between 4th and 6th grade students is evident only for those with higher academic achievements. Only 4% of the weak students in grade 4 were able to answer the question in which they were requested to write a simple program (entering two numbers into variables and printing their sum). Only 6th grade students with high achievements did well on this question (89% correct); all the others did less well (30-40% correct answers).

The Final Test

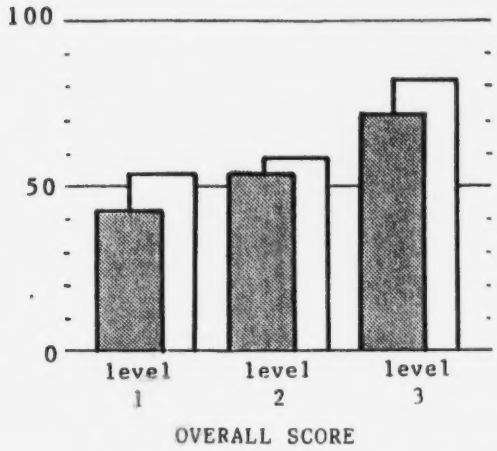
The students' scores in the final test are the best available indication for their understanding of the concepts at the completion of the learning process.

Table 4.3 shows the mean scores in the final test, in percents, divided according to grade level and academic achievements. The first four sections give the scores on each concept separately, and the overall score appears in the fifth section. Numbers in parentheses indicate the number of students. Figure 2 presents the same data graphically.

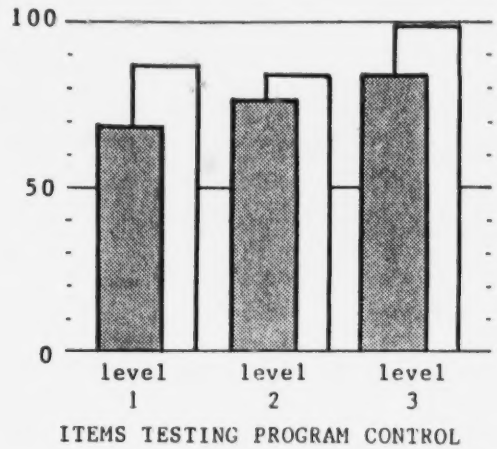
TABLE 4.3: THE MEAN SCORES IN THE 5 FINAL TESTS ADMINISTERED IN STUDY 2

	Graphic statements (4 items)				Random number (2 items)				Loop (3 items)				Variable (4 items)				Overall score (15 items)			
grade level	1	2	3	ALL	1	2	3	ALL	1	2	3	ALL	1	2	3	ALL	1	2	3	ALL
4	69	77	84	77	21	32	65	47	50	56	72	61	20	33	59	41	43 (12)	54 (12)	72 (19)	59 (43)
6	87	84	99	90	31	50	75	53	64	68	79	70	20	28	69	39	54 (6)	59 (8)	82 (7)	65 (21)
all				81				49				64				40				61 (64)

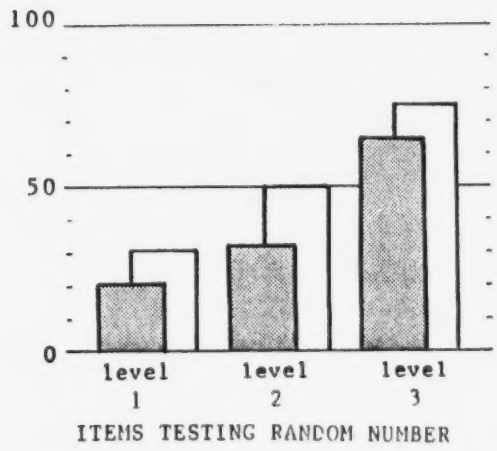
FIG. 2: MEAN SCORES IN THE FINAL TEST (STUDY 2)



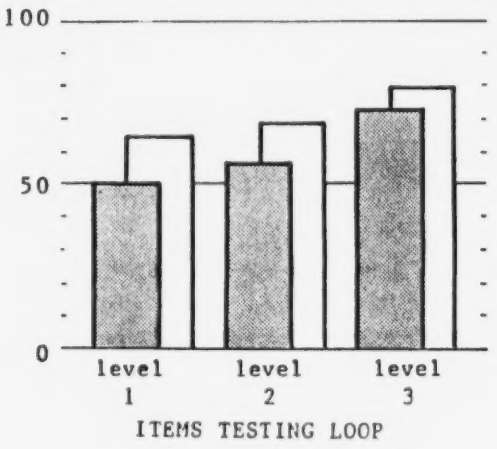
OVERALL SCORE



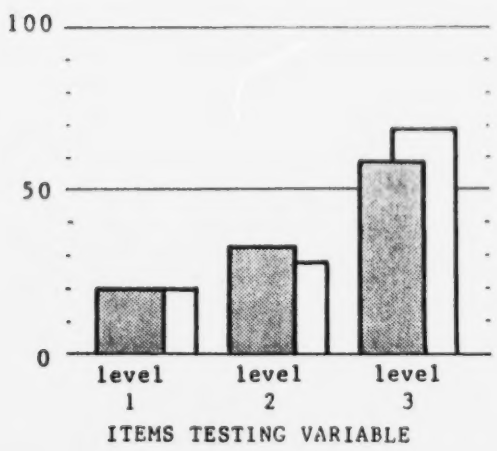
ITEMS TESTING PROGRAM CONTROL



ITEMS TESTING RANDOM NUMBER



ITEMS TESTING LOOP



ITEMS TESTING VARIABLE

LEGEND

4th GRADE

6th GRADE

The same trends which were apparent in the earlier tests, are noted in the final test scores.

Most students showed a good ability of writing a simple graphic program, and an ability to change a program and work with it. Children with average and poor academic achievements demonstrated especially poor achievements in understanding the random number and variable and related concepts. This was in contrast with the high achievements of those students in both grade levels, who had high academic achievements. Understanding the loop concept was average, and the achievements were improved, as compared to scores achieved in the intermediate tests.

The sources of variation

A three-way ANOVA was conducted, in an attempt to examine the sources of high variability in the achievements of the students. The effects of grade level (age), academic achievements level and sex, were examined.

Table 4.4 presents the results of ANOVA of the total score in the intermediate and final tests.

Table 4.4: SOURCE OF VARIANCE

MAIN EFFECTS	grade			level of academic studies			sex		
Concepts	F	D.F.	SIGNIF	F	D.F	SIGNIF	F	D.F.	SIGNIF
Graphic statements	5.9	1,59	.018 *	3.4	2,59	.041 *	0.0	1,59	.90
Programs	17.5	1,46	.001**	3.6	2,46	.035 *	0.1	1,46	.81
Random no.	1.1	1,58	.299 *	6.7	2,58	.002**	0.2	1,58	.65
Loop	1.6	1,58	.209	1.0	2,58	.391	1.6	1,58	.12
Variable	3.6	1,47	.063	11.6	2,47	.001**	0.7	1,47	.39
Final test	1.7	1,62	.202	7.8	2,52	.001**	0.0	1,52	.99

* $P < .05$

** $P < .01$

Examination of the data reveals that the main sources of variability are academic level and age. The student's sex had no effect whatsoever on achievements. Second- and third-order interactions had no significant effects.

Attitudes

At the completion of instruction, the students completed an attitude questionnaire, asking about their opinions of the course, and programming in general. The following three tables present the replies (in percents). This does not constitute a thorough examination of the students' attitudes, but rather an impression of their opinions regarding the course they had taken.

TABLE 4.5: DID YOU ENJOY THE COURSE? (N=64)

	very much	in general	very little	did not enjoy	
grade 4	59	39	2	0	100%
grade 6	81	19	0	0	100%

TABLE 4.6: WOULD YOU LIKE TO CONTINUE STUDYING PROGRAMMING? (N=64)

	yes, very much	I think so	not sure	no	
grade 4	46	27	25	2	100%
grade 6	62	29	9	0	100%

TABLE 4.7: DO YOU THINK THAT PROGRAMMING SHOULD BE TAUGHT IN SCHOOL? (N=64)

	yes	no	don't know	
grade 4	70	7	23	100%
grade 6	81	0	19	100%

Most students enjoyed the course and wanted to continue studying this subject. The 6th grade

students consistently demonstrate a more positive response, as compared to the 4th grade students.

The children were given opportunity to explain their response to the question regarding teaching programming in school. Positive responses appeared to result from the interest and pleasure of learning and interacting with the computer. Most reservations and objections were explained by the child's conviction that the standard of teaching in school would not be satisfactory, and one mustn't spoil the pleasure derived from this learning.

Table 4.8 presents the responses to the question regarding the degree of difficulty of the course. The possiblities among which the students had to choose did not permit avoiding the expression of opinion. It is interesting to note that about two-thirds of the 4th grade students had difficulties some of the time, and only one third of them found the course easy. For the 6th grade students, the situation is the other way round: Only a negligible proportion of the children found all lessons either too difficult or too easy.

TABLE 4.8: WAS IT TOO DIFFICULT OR TOO EASY TO LEARN PROGRAMMING? (N=64)

	always too hard	sometimes hard	usually easy	always too easy	
grade 4	2	64	32	2	100%
grade 6	0	29	71	0	100%

5. ANALYSIS OF THE RESULTS

The results of the present study suggest that children in elementary school are mature enough and can learn basic concepts of programming language taught with an adequate curriculum. Major components in adjusting the teaching material to naive learners in general, and young children in particular, include: hierarchical presentation of concepts; presentation of the variable at a later stage of learning; Hand-on experience with a micro-computer; wide use of graphics; and disconnection of the learning from mathematical contents.

Most learners reached the level of writing a simple program, comprehesion of the management of the program, and basic understanding of some of the concepts presented (such as simple action commands and computer graphics). However, rather poor achievements were observed with regard to the acquisition of more abstract concepts, resulting in a relatively high number of errors related to the understanding of concepts such as loop and variable.

Focal points of difficulty in learning:

The following focal points of difficulty were observed during learning:

Multiple-parameter statements proved far more difficult than single-parameter statements, or statements without any parameters at all.

Infinite loop incurred misconceptions on many occasions. The major difficulty appears to lie in the additional complexity in the flow of the program (loop, instead of a linear program). This new structure presents difficulties both in understanding its role in the domain of the program's flow, and in integrating it with other knowledge domains (e.g. random or consistent changing of parameters). Interestingly, it was the operational definition of the infinity of the loop (the program should go on and on, unless we stop it with the keyboard) which was properly understood.

The variable concept is major to the learning of a programming language; it comprises one of the major difficulties for children, especially the younger ones. Changing values according to a fixed

pattern (eg. X=X+1), as well as logical questions involving variables (eg. IF X>8), incur misunderstanding which may be the result of lack of maturation of these children with respect to the comprehension of formal concepts.

The distinction between a situation in which the use of an input statement stops the program, to await inputting by the user, and the situation in which the program is being edited, was particularly hard to grasp. This distinction is an essential component of the interactive nature of the program, and therefore deserves further examination.

Comprehension of the conditioning also appeared to involve great difficulties, resulting from misunderstanding of the logic operation itself, as well as confusion with regard to the use of condition statements for branching a program.

Levels of comprehension and application

Examination of the work of the students suggests that the acquisition of each concept involves a number of levels of comprehension, and this is reflected in the ability to apply the concept.

It appears that the comprehension of a concept may be divided into the following levels:

1. Literal understanding of the idea at the basis of the concept.
2. Understanding the syntax of the statements related to the concept.
3. Understanding the statement applying the concept.
4. Understanding the execution of a simple program which applies the concept.
5. Ability to introduce a small change into a given program, in order to induce a limited change in the result.
6. Ability to write a simple program according to a given pattern.
7. Ability to write a simple program exceeding the above pattern, in order to solve a specific problem.
8. Ability to integrate the concept in another program, involving the application of other concepts.

Throughout the learning, children were observed to experience most difficulties in stages 5-8,

involving the application of the acquired knowledge to writing an independent program (or part of a program). The first four stages appeared to cause much less misunderstanding and confusion.

Sources of variability

The students' achievements reveal much variability in the acquisition of various concepts. On many occasions (for example, in understanding the variable concept) some students reached the highest level, whereas others were left with total incomprehension. Nevertheless, it was most impressive to note that throughout the learning, each student was deeply involved in learning, at the specific level which was suitable for him.The variability among the students may be analyzed according to the following:

Differences between 4th grade students and 6th grade students:

Generally speaking, differences between the two age groups were observed almost in every stage, with respect to each concept, as well as at the final test. The most significant difference was with regard to the variable concept. Whereas the 6th grade students (and particularly the better ones) succeeded in understanding this concept, the great majority of 4th grade students completely misunderstood it. In Study 1, a significant decrease in the achievements of the 4th grade students coincided with the introduction of the variable concept, which was followed by the input and condition variables, resulting in rather poor understanding of the latter concepts.

Another finding noted in the final test of Study 1 is the greater difficulty encountered by the 4th grade students, as compared to the 6th grade students, in applying their knowledge when writing an original program.

The acquisition of the concepts took longer for the younger children. This is reflected by the significant differences between the two groups during the learning, which grew smaller (even though they did not disappear entirely) at the final test.

Differences between students with high, intermediate and low academic achievements:

It is hardly surprising that students with high academic achievements, excelled in programming as

well, as compared to those with poor academic achievments. It should be noted that the poor students, especially in the 4th grade, had very poor achievements in programming. This finding is evident in Study 2, both in the intermediate and in the final tests. The gap between students is larger when abstract concepts, such as variable and random number, are introduced. Parenthetically it should be mentioned that most 6th grade students who were excluded from the experiment because of previous knowledge in programming, were students with high academic achievements (21 of 23 children).

Sex differences:

The present discussion should be separated into two questions: Firstly, is there any difference between the sexes with regard to the curiosity and motivation for studying programming, as expressed in the effort invested in finding an informal setting for learning; and secondly, when they are already within a learning setting (such as in Study 2), are there any differences in achievements between boys and girls?

The present results are unequivocal with regard to both questions. In the 4th grade, all children with previous knowledge of programming were boys; 91% of those with previous knowledge at the 6th grade, were also boys. Thus, the answer to the first question appears to be positive: In the studied population, programming is considered an activity for boys, and only few girls become involved in it.

As for the second question: here, too, the present study supplies a clear-cut answer: No differences whatsoever were noted between boys and girls who participated in the study. Moreover, the ANOVA clearly indicates that the sex variable has no effect on the achievements of the learners in either comprehension or application of any of the concepts which were learned.

6. CONCLUSIONS

The present study has demonstrated the complexity of investigating the acquisition of a programming language by children. The potential advantages of introducing the teaching of a programming language as a discipline, are accompanied by evident difficulties in learning and comprehension of basic concepts. Evidently, much curricular work and thorough cognitive research are required.

The importance of equipping each student with basic technological-scientific literacy, as an essential part of his basic education, does not permit that programming teaching be relinquished, left to be taken care of by some bodies with commercial interests, neighbouring on the borders of the informal education system. One measure of preventing further increase in the gap among various classes and the formation of a technological elite to which a small minority of the weaker classes may belong, is introducing the computer programming to the formal education system.

The large variability in achievements, and the numerous misconceptions noted among some children during the learning process, emphasize the need for a comprehensive, detailed descriptive model of acquisition of a programming language by novices, as well as finding satisfactory curricular solutions for teaching abstract and difficult concepts. The need for more detailed studies regarding children's perception of specific concepts of programming, such as "loop" and "variable", is evident.

The present results do not support the common belief among educators, suggesting that students with poor academic achievements may express themselves more successfully through programming, thus perhaps breaking through the vicious circle of failure. To the contrary, children whose academic achievements were poor, also demonstrated poor understanding of basic concepts of programming language. This may point to another possible difficulty when attempting to apply programming teaching in schools, on a wider scale.

Our results suggest that there should be no differences between boys and girls, when programming is taught at schools. Rather, teaching programming in school may contribute to the increasing number of girls who

successfully learn programming.

Computer programming is a time-consuming activity. Learning the information related to each of the concepts, extending the areas of application and the level of comprehension, require much time for learning and practice. Planning the goal design for teaching programming in school must, therefore, take into account the relatively long time for learning and practice, necessary for the naive learner. If the purpose of teaching is that the students become programmers capable of directly employing a programming language in order to solve simple problems, one must allocate to this teaching much time during the higher grades of the elementary school and all through high school. If, however, the education system cannot afford but allocate more limited time for programming teaching, one must be more modest with regard to both the goals and the contribution of programming teaching at school. This is in agreement with the conclusions of others (Kurland, Mawby and Cahir, 1984).

Children master the basic interaction with the computer naturally and quickly, unlike some adults. This indicates that programming teaching may be introduced at the level of the upper grades of elementary school. The relative success of students in the 4th and 6th grades, when studying programming without variables, supports the above suggestion. However, the difficulties encountered by practically all 4th grade, and most of the 6th grade students, in mastering abstract concepts such as variable (which are central to the process of programming), point to the need to postpone the teaching of these concepts to a later stage of learning. This need was expressed by the principles of the described curriculum, and stated in Luehrmann's conclusions (Luehrmann, 1983). The present findings also support Becker's conclusions, stating that students reach the maturity required for understanding the abstract concepts related to programming only in grades 6 to 8. At this age they depend less on concrete actions for understanding the programming, while they are still young enough to have the open mind which is an asset when coping with an entirely new activity (Becker, 1982).

Comprehension of the concepts with which students had difficulties in the present study, may be aided by concrete models such as those described by Mayer (1981), and interactive programs illustrating the sequence in which statements are being executed in a computer program, and concurrently presenting the program, the

output and the variable values. Software such as "The transparent computer" (Hioduser, Nachmias & Chen, 1984), "Spider world" (Dalbey & Linn, 1984) or "Karl the robot" (Pattis, 1981) may serve as important auxilliaries in teaching principles and basic concepts of programming. They may be instrumental in avoiding the need to cope with syntax problems involved in operating an ordinary programming language, thus permitting to devote the required time for learning a programming language (Hioduser, Nachmias & Chen, 1985).

Many educators point to the importance of teaching the principles of structured programming and the top down design (Papert, 1980), when teaching programming. This issue is the essential difference between LOGO and BASIC, as computer languages. The importance of employing structured programming in relation to professional programming and software development is evident. However, teaching the principles of structured programming may not necessarily be relevant to the teaching of introductory programming to children. In the presnet study, the children were involved in writing programs with a relatively simple structure, and did not require structuring techniques. Likewise, the majority of children learning LOGO managed to master only the basic turtle graphics commands and simple aspects of the program ontrol structure, and did not benefit from the advantages of structured programming (Kurland, Clement, Mawby & Pea, 1984).
The above findings indicate that a thorough study of the relevance of procedural thinking to the thought processes of children is required.

The graphic medium, through which new concepts were presented, proved most successful. This medium enables much more varied range of programming activities, and adjusts them to the child's experience. The work of children of all ages in Cartesian co-ordinates was facile. In view of the frequent use of the turtle geometry as the graphic medium of programming language (Abelson & diSessa, 1980), the need for a comparative study arises. Such a study should look at the application of computerized drawing, graphics and animation, with Euclid, as compared to turtle, geometry.

Teaching a programming language in school is a true challenge. We must ensure that programming is presented as a thought-provoking, interesting and creative activity, suitable for as wide as possible a range of learners. This may perhaps prevent the emergence of the illiteracy of the computer age - the inability to use the computer.

7.REFERENCES

Ableson, H.; diSessa, A., Turtle Geometry , Cambridge, MA; MIT Press, 1980.

Becker, J.H., "Microcomputers In The Classroom - Dreams and Realities", N.I.E. Report No. 319 , USA, 1982.

Dalbey J.; Linn M., "Spider Worlds: A Robot Language for Learning to Program", ACCCEL Research Report, University of California at Berkley, 1984.

diSessa, A., "A Principled Design for an Integrated Computational Environment", Human Computer Interaction, Vol.1 No.1 , 1984.

Kurland, D.M.; Clement, K.; Mawby, R.; Pea, R.D., "Mapping the Cognitive Demands of Learning to Program", in: Kurland, M.D.(ed.), Developmental Studies of Computer Programming Skills , Paper Presented in AERA Annual Meeting, 1984.

Kurland, D.M.; Mawby, R.; Cahir, N. "The Development of Programming Expertise in Adults and Children " in: Kurland, M.D.(ed.), Developmental Studies of Computer Programming Skills , Paper Presented in AERA Annual Meeting, 1984.

Luehrmann, A., "Slicing Through Spaghetti Code", The Computer Teacher , April 1983.

Luehrmann, A.; Peckham, H., Computer Literacy, A Hands-On Approach , McGraw-Hill Book Company, USA, 1984.

Mioduser, D.; Nachmias, R.; Blau, A.; Chen, D., The Transparent Computer , Student Handbook, The Computers in Education Research Lab., Tel-Aviv University, Israel, 1984.

Mioduser, D.; Nachmias, R.; Chen, D., "Teaching Programming Literacy for Non-Programmers", submitted for publication, 1985.

Nachmias, R.; Mioduser, D.; Chen, D., "A Close Look at Children's Acquisition of Computer Language: Six Case Studies", The Computers in Education Research Lab., Research Report No. 5 (Hebrew), Tel-Aviv University, Israel, 1984.

Nachmias, R.; Mioduser, D.; Chen, D., Programming is a Child's Game (Hebrew), Masada, in press, Israel, 1985.

Nachmias, R.; Mioduser, D.; Chen, D., "Teaching Programming at School: A 4th and 6th Grade Experience", The Computer in Education Research Lab., Research Report No. 6 (Hebrew), Tel-Aviv University, Israel, 1985(a).

Mayer, R.E., "A Psychology of Learning BASIC", Communication of the ACM, 22 , 1979, 589-593.

Mayer, R.E., "The Psychology of Learning Computer Programming by Novices", Computing Surveys, 13 , 1981, 121-141.

Moshell, M., Computer Power: A First Course in Using the Computer , McGraw-Hill, the University of Tennessee, 1982.

Papert, S., Mindstorms: Children, Computers, and Powerfull Ideas , Basic Books, Inc., New-York, 1980.

Pattis, R., Karl The Robot , Wiley & Sons, New York, 1981.

Pea,R.D.; Kurland, D.M., "Logo Programming and the Development of Planning Skills", Bank Street College of Education, Technical Report No. 16 , 1984.

Shell, B.A., "Teaching Procedural Literacy", in Proceedings of The ACM Annual Conference , Nashville, Tennessee, 1980, 125-126.

Shell, B.A., "Coping with complexity", Proceeding of Houston Symposium III, Information and Society , 1980(a).

Shell, B.A., "The Psychological Study of Programming", Computing Surveys, 13 , 1981, 101-120.

Shneiderman, B., "Teaching Programming: A Spiral Approach to Syntax and Semantics", Computers and Education, 1 , 1977, 193-197.

Shneiderman, B., Software Psychology: Human Factors in Computer and Information Systems , Winthrop, Cambridge, MA., 1980.

Weinberg, G.M., The Psychology of Computer Programming, Van Nostrand Reinhold, New-York, 1971.